B504/I538: Introduction to Cryptography

Spring 2017  •  Lecture 12

(2017-02-16)
Strong end-to-end encryption is legal in the United States today, thanks to our victory in what’s come to be known as the Crypto Wars of the 1990s. But in the wake of Paris, San Bernardino, and Orlando, there is increasing pressure from law enforcement and policy makers, both here and abroad, to mandate so-called backdoors in encryption products.

In this presentation, I will discuss in brief the history of the first Crypto Wars, and the state of the law coming into the Trump Era. I will then discuss what happened in the fight between Apple and the FBI in San Bernardino and the current proposals to weaken or ban encryption, covering proposed and recently enacted laws in New York, California, Australia, India, and the UK. Finally, I will discuss possible realistic outcomes to the Second Crypto Wars, and give my predictions on what the State of the Law will be at the end of 2017.

Nate Cardozo is a Senior Staff Attorney on the Electronic Frontier Foundation’s digital civil liberties team. In addition to his focus on free speech and privacy litigation, Nate works on EFF’s security and crypto policy and the Coders’ Rights Project. Nate has projects involving export controls on software, state-sponsored malware, automotive privacy, government transparency, hardware hacking rights, anonymous speech, electronic privacy law reform, and Freedom of Information Act litigation.

A 2009-2010 EFF Open Government Legal Fellow, Nate has a B.A. in Anthropology and Politics from U.C. Santa Cruz and a J.D. from U.C. Hastings.
Recall: MAC existential forgery game

Let $E$ be the event that $(m, t) \notin \{(m_1, t_1), \ldots, (m_n, t_n)\}$ yet $Ver_k(m, t) = 1$.

Define $A$'s advantage to be $\text{Adv}^{\text{MAC-strong-ex-forgery}}(A) := \Pr[E]$. 

Ryan Henry
Recall: Naïve CBC-MAC

- Let $\{f_k\}_{k \in \{0,1\}^*}$ be a PRF family
- $\text{Gen}(1^n)$ outputs a uniform random key $k \in \{0,1\}^n$
- $\text{MAC}_k(m)$ does the following:
  1. Split $m$ into $n$-bit blocks $m_1, \ldots, m_n$
  2. Initialize $t_0 = \{0\}^n$
  3. Compute $t_i = F_k(t_{i-1} \oplus m_i)$
  4. Output the tag $t := t_n$
- $\text{Ver}_k(m, t)$ outputs 1 if $t = \text{MAC}_k(m)$ and 0 otherwise
Recall: Naïve CBC-MAC

$m = m_1 || m_2 || \cdots || m_\ell$
Recall: Attacking naïve CBC-MAC

Challenger (C)

1^n

k ← Gen(1^n)

t ← MAC_k(m)

Forger (A)

m ∈ \{0, 1\}^n

m' := m || (m ⊕ t)

A's output is a valid forgery because F_k(m') = F_k((m ⊕ t) ⊕ t) = F_k(m) = t

Ryan Henry
CBC-MAC fix #1: Prepend the block-length

$m = m_1 || m_2 || \ldots || m_l$

Intuitively, MAC on $n$-block message is useless for forging MACs on $n'$-block messages.
CBC-MAC fix #2: Length-specific key

\[ m = m_1 || m_2 || \ldots || m_\ell \]

\[ k_\ell = F_k(\text{block length}) \]

(input padded to block size)

Again, MAC on \( n \)-block message is useless for forging MACs on \( n' \)-block messages
CBC-MAC fix #3: Nested CBC-MAC (NMAC)

\[ m = m_1 \| m_2 \| \cdots \| m_\ell \]

\[ k_\ell = F_k(\text{block length}) \]
(input padded to block size)

Compute Naïve CBC-MAC with first key.

MAC the Naïve CBC-MAC with second key.

Ryan Henry
CBC-MAC versus CBC mode encryption

- CBC mode encryption requires uniform random IV
  - Otherwise, it is not IND-CPA secure!
- CBC-MAC requires fixed IV
  - Otherwise, it is not existentially unforgeable!
- CBC mode encryption outputs each block
  - Otherwise, it is not correct!
- CBC-MAC only outputs a single block (the last one)
  - Otherwise, it is not existentially unforgeable!
- CBC mode encryption requires a PRP
  - Otherwise, it is not correct!
- CBC-MAC only requires a PRF
Hash functions

**Defn:** A hash function is a PPT function \( H : \{0, 1\}^* \rightarrow \{0, 1\}^s \) that maps arbitrary-length bit strings into fixed-length bit strings.

The output of a hash function is called a "hash", "digest", or "fingerprint" of the input.
Hash function collisions

**Definition:** Let $H$ be a function taking on values in $\{0, 1\}^*$. A collision for $H$ is an ordered pair $(m_0, m_1) \in \{0, 1\}^*$ of distinct inputs such that $H(m_0) = H(m_1)$.

Pigeon-hole principle: If the domain of $H$ is (much) larger than its range, then (many) collisions must exist!

more pigeons $\rightarrow$ more collisions
Collision resistance

- Intuitively, we want to say that no PPT algorithm can find a collision for $H$, except with a probability that is negligible in $s$ (the length of the output).

Q: How do we formalize this notion?

A: Very carefully...

  - Difficulty: once $H$ is fixed, it is trivial to define a PPT algorithm that has a collision for $H$ “hard-coded”.
Keyed hash functions

**Def**: A keyed hash function with output length $\ell(s)$ is a pair of PPT algorithms $(Gen, H)$ such that

- $Gen(1^s)$ outputs a uniform random key in $k \in \{0, 1\}^s$
- $H(k, x)$ outputs a fingerprint $y \in \{0, 1\}^{\ell(k)}$

$x \in \{0, 1\}^*$

**Idea**: Define collision resistance to require that no PPT algorithm can find a collision for $H$ when the key is selected at random, except with probability negligible in $s$. 
Let $E$ be the event that $m_0 \neq m_1$ and $H(k, m_0) = H(k, m_1)$

Define $A$'s advantage to be $\text{Adv}_{\text{collision}}(A) := \Pr[E]$

Defn: A keyed hash function $(\text{Gen}, H)$ is collision resistant if, for every PPT attacker $A$, there exists a negligible function $\varepsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\text{Adv}_{\text{collision}}(A) \leq \varepsilon(s)$. 
Second preimage resistance

a.k.a target collision resistance

Informally, a keyed hash function \((\text{Gen}, H)\) is second preimage resistant if no PPT attacker can, given \(m_0 \in \{0, 1\}^*\) and \(k \leftarrow \text{Gen}(1^s)\), output \(m_1 \in \{0, 1\}^*\) such that \(m_0 \neq m_1\) and \(H(k, m_0) = H(k, m_1)\) except with probability negligible in \(s\).
Second-preimage-finding game

Let $E$ be the event that $m_0 \neq m_1$ and $H(k, m_0) = H(k, m_1)$

Define A's advantage to be $\text{Adv}_{2\text{-preimage}}(A) := \Pr[E]$

**Defn:** A keyed hash function $(\text{Gen}, H)$ is second preimage resistant if, for every PPT attacker $A$, there exists a negligible function $\varepsilon: \mathbb{N} \to \mathbb{R}^+$ such that $\text{Adv}_{2\text{-preimage}}(A) \leq \varepsilon(s)$. 
Second preimage resistance

Thm: If \((\text{Gen}, H)\) is collision resistant, then it is also second preimage resistant.

Proof: Just note that a second preimage is a collision.

Q: Is the converse of this theorem true?
A: No! (But why?)
Preimage resistance
a.k.a. one-wayness

- Informally, a keyed hash function $(Gen, H)$ is preimage resistant if no PPT attacker can, given $k \leftarrow Gen(\lambda^s)$ and $y \in \{0, 1\}^{\ell(s)}$, output $m \in \{0, 1\}^\ast$ such that $H(k, m) = y$ except with probability negligible in $s$. 
**Preimage-finding game**

Let $E$ be the event that $H(k, m) = y$

Define $A$’s advantage to be $\text{Adv}_{\text{preimage}}(A) := \Pr[E]$

**Defn**: A keyed hash function $(\text{Gen}, H)$ is preimage resistant if, for every PPT attacker $A$, there exists a negligible function $\varepsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\text{Adv}_{\text{preimage}}(A) \leq \varepsilon(s)$. 

Challenger (C)

$k \leftarrow \text{Gen}(1^n)$

Attacker (A)

$1^n, y \in \{0, 1\}^{\ell(n)}$

$m$
Preimage resistance

Thm: If \((\text{Gen}, H)\) is preimage resistant for randomly selected inputs, then it is also second preimage resistant.

Proof (sketch): Suppose that \(A\) breaks preimage resistance.
- Given \(k\) and \(m\), compute \(y = H(k, m)\)
- Now use \(A\) to find a preimage of \(y\).
- Since \(y\) has many preimages, with high probability that preimage that \(A\) finds will not be \(m\)!

Q: Is the converse of this theorem true?
A: No! (But why?)
(One-way) compression functions

- Intuitively, a (one-way) compression function is a keyed function $h$ with three properties:
  - Efficient: There exists a PPT algorithm that evaluates $h$
  - Compression: $h$ maps $2s$-bit strings to $s$-bit strings
  - One-way: Given an output of $h$, it is difficult to find any input that maps to that output
    (This is the opposite of what you want from non-cryptographic compression functions)

Q: On average, how many inputs map to each output?  
A: About $2^s$
Merkle-Damgård construction

\[ m := m_1 m_2 \ldots m_n \]

\[ h_k(m) := 0^n \]

\[ h_k(m_1) \longrightarrow h_k(m_2) \longrightarrow h_k(m_n) \]

\[ H_k(m) := z_n \]
Davies-Meyer compression function

Thm: If $F$ is a PRF, then the Davies-Meyer compression function is collision resistant. In particular, finding a collision requires $O(2^{n/2})$ evaluations of $F$ on average.

$$Z_i := F_{m_i}(z_{i-1}) \oplus z_{i-1}$$
Recall: Nested CBC-MAC (NMAC)

\[ m := m_1 m_2 \ldots m_n \]

Compute Naive CBC-MAC with first key $\text{MAC} \text{Naive CBC-MAC with second key}$
Hash-based MAC (HMAC)

- The most widely used MAC algorithm in practice

\[
\text{HMAC}_{s,k}(m) := H_s((k \oplus \text{opad}) \mathbin\| H_s((k \oplus \text{ipad}) \mathbin\| m))
\]

- \(H_s\) is a collision-resistant (keyed) hash function
- \(k\) is the secret MAC key
- \(\text{opad} = 0x5c5c5c \ldots 5c\)
- \(\text{ipad} = 0x363636 \ldots 36\)

\(\text{opad}\) and \(\text{ipad}\) are chosen so that \((\text{opad} \oplus \text{ipad})\) has a large Hamming weight
HMAC

\[ m := m_1 m_2 \ldots m_n \]

\[ k \oplus \text{ipad} \rightarrow h_s \]

\[ m_1 \rightarrow h_s \rightarrow m_n \rightarrow h_s \]

\[ k \oplus \text{opad} \rightarrow 0^s \rightarrow h_s \rightarrow 0^s \rightarrow h_s \]

\[ t \]
Simpler HMAC constructions?

Q: Is \( H(k \| m) \) a secure MAC?
A: No! (But why?)
- Suppose \( H \) is constructed using Merkle-Damgård construction
- Given \((m, H(k \| m))\) it is easy to compute \( m' := m \| m'' \) and \( t' \) such that \( t' = H(k \| m')! \) (But how?)
- Just set \( t' = H(t \| m'' \)

Q: Is \( H(m \| k) \) a secure MAC?
A: Errr, well....sort of!? It's not as secure as HMAC! (But why?)
- If \( H(m_0) = H(m_1) \) then \( H(m_0 \| k) = H(m_1 \| k) \)
- Weakness in collision-resistance of \( H \) implies weakness in HMAC
Simpler HMAC constructions?

Q: Is $H(k \| m \| k)$ a secure HMAC?

A: I don't know! Possibly?

- This is essentially HMAC without $ipad$ and $opad$
- Proof of existential unforgeability for HMAC requires that $ipad$ and $opad$ differ in at least one bit!
- $H(k \| m \| k)$ falls to "target prefix collision" attacks against $H$
Generic birthday attack

Let \( H : \{0, 1\}^* \rightarrow \{0, 1\}^s \) and consider the following algorithm:
- Choose \( N := (5/4) \cdot 2^{s/2} \) distinct messages, \( m_1, \ldots, m_N \), each uniformly at random
- For \( i = 1, \ldots, N \), compute \( y_i := H(m_i) \)
- If \( y_i = y_j \) for some \( i \neq j \), then output \( (m_i, m_j) \)

Thm (birthday paradox): Let \( r_1, \ldots, r_N \) be independently and identically distributed random variables taking on values in \( \{0, 1\}^s \)
- If \( N = (5/4) \cdot 2^{s/2} \) then \( \Pr[ \exists i \neq j, r_i = r_j ] \geq 1/2 \).
Generic birthday attack

Thm (birthday paradox): Let $r_1, \ldots, r_N$ be independently and identically distributed random variables taking on values in $\{0, 1\}^s$.

If $N = (5/4) \cdot 2^{s/2}$, then $\Pr[\exists i \neq j, r_i = r_j] > 1/2$.

Proof (for uniform random variables):

\[
\Pr[\exists i \neq j, r_i = r_j] = 1 - \Pr[\forall i \neq j, r_i \neq r_j] = 1 - (\frac{2^s-1}{2^s}) (\frac{2^s-2}{2^s}) \cdots (\frac{2^s-N+1}{2^s}) = 1 - \prod_{i=1}^{n-1} (1 - i/2^s) \geq 1 - \prod_{i=1}^{n-1} e^{-i/2^s} = 1 - e^{-\frac{N^2}{2}/2^s} \geq 1 - e^{-\frac{(5/4 \cdot 2^{s/2}/2)}{2^s}} = 1 - e^{-25/32} \geq 0.54
\]
Generic birthday attack

• Obs: An attacker $A$ that uses the generic birthday attack can find collisions with advantage $\text{Adv}_{\text{collision}}(A) > 1/2$ in $O(s \cdot 2^{s/2})$ time (albeit with $O(s \cdot 2^{s/2})$ storage)

Q: Is this a problem?
A: No! (in theory); Possibly! (in practice)
  - Real hash functions have fixed-length outputs
  - Need to ensure that $2^{s/2}$ work is infeasible... or do we? Memory is scarcer than time

Q: Is it sufficient to ensure no real attacker can store $s \cdot 2^{s/2}$ bits?
A: Perhaps surprisingly, no!
"Small-space" birthday attack

Consider an attacker A that works as follows:
1. Choose a random initial value $m_0$
2. Set $m := m_0$ and $m' := m_0$
3. For $i = 1, 2, 3, \ldots$, do the following
   - Compute $m := H(m)$ and $m' := H(H(m'))$  \(\text{// now } m = H^i(m_0) \text{ and } m' = H^{2i}(m_0)\)
   - If $m = m'$, break from loop
4. Set $m' := m$ and $m := m_0$
5. For $j = 1, \ldots, i$, do the following
   - If $H(m) = H(m')$, return $(m, m')$
   - Else, set $m := H(m)$ and $m' := H(m')$  \(\text{// now } m = H^j(m_0) \text{ and } m' = H^{i+j}(m_0)\)

Thm: The above small-space birthday attack finds a collision with probability at least $1/2$ in $O(s \cdot 2^{s/2})$ time using $O(1)$ storage.
That's all for today, folks!